Amendments to the Specification:

Please amend the paragraph beginning at page 3, line 6 as follows:

A system, method and computer readable medium containing programming instructions ~~structions~~ for an improved latch mechanism for use in a programming environment running concurrent tasks is disclosed. According to one aspect of the present invention, a latch is provided to a first task which is requesting access to a resource. After such access, the first task holds the latch, which is also released from the first task. The first task holding the latch marks the latch stealable, and if the latch is not marked stolen by another task, the first task can mark the latch unstealable. In another embodiment of the present invention, if the first task is provided a latch marked stealable, the resource associated with the stealable latch will be placed in a consistent state before the first task accesses the resource.

Please amend the paragraph beginning at page 7, line 11 as follows:

In terms of the example of Figure 1, if task 10 holds the latch but has marked it STEALABLE and if task 10 again seeks to execute code 16, task 10 must first call the function mark_unstealable(). If mark_unstealable() returns with a value to indicate that the holder's global flag for that latch is STEALABLE then task 10 may execute code 16 to access the latched resource. Thus task 10 may avoid executing cleanup subroutine 14 between successive accesses to the resource made by code ~~18~~ 16.

Please amend the paragraph beginning at page 7, line 17 as follows:

The second possible event referred to above occurs where a second task requests the exclusive latch before the holding task reaccesses the resource (by calling ~~mark_stealable()~~ mark_unstealable()). In this case, the second task (task 12 in the example of Figure 1), will seek to gain access to the latch in the usual manner, using the latch() function. In the preferred embodiment, the latch() function will provide the latch to the second task if it is available. The latch is available if it is not held by another task or if it is held by another task in a mode compatible with the mode in which the second task is seeking the latch.